C++ SESSION 2

FUNCTIONS AND TIME COMPLEXITY

SESSION'S AGENDA

- What's a function?
- Why we use functions?
- Pre-defined or built-in functions.
- User-defined functions.
- Passing by value.
- Exercises.
- Algorithm analysis and time complexity

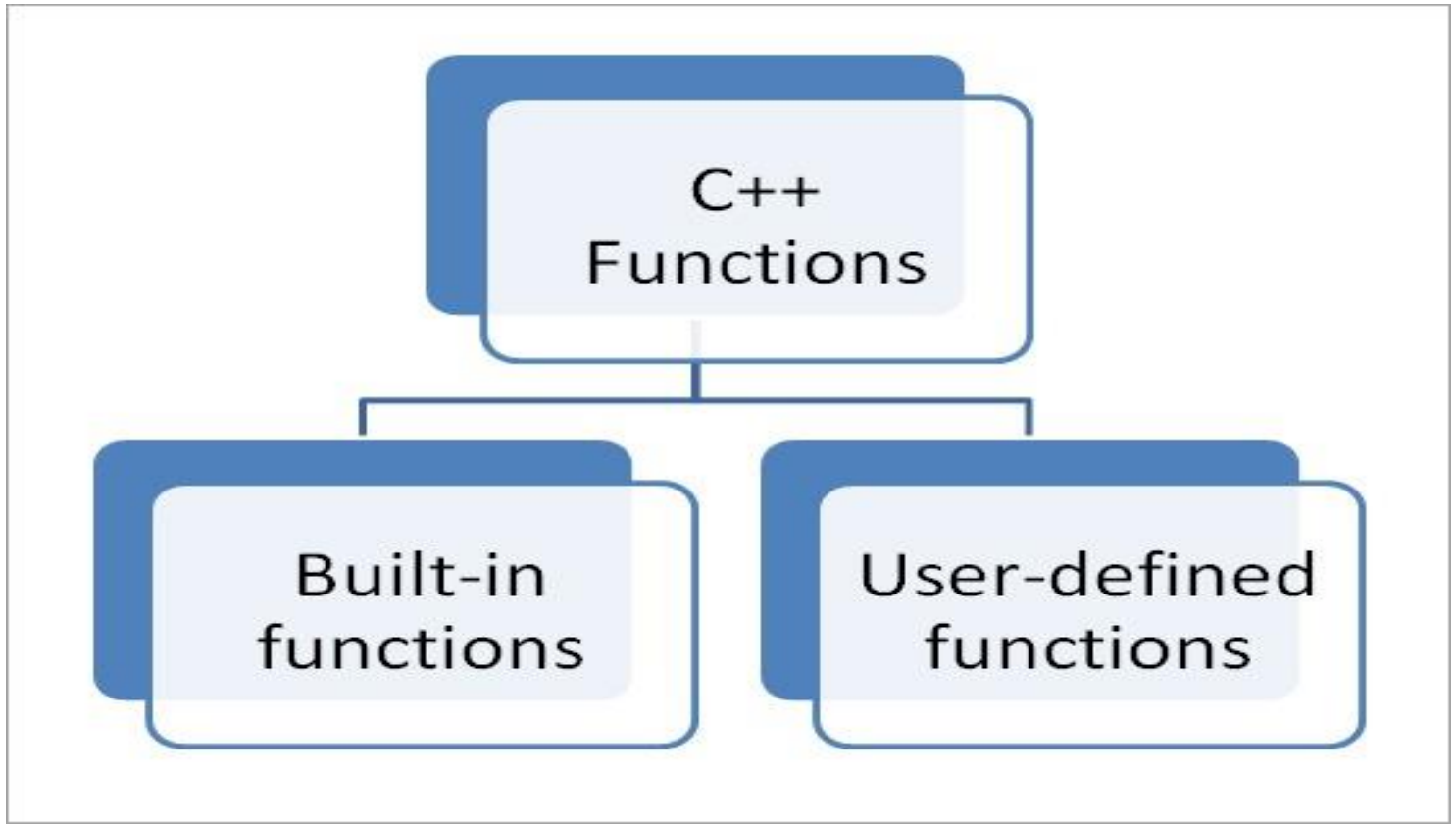
WHAT'S A FUNCTION? (20 MINUTES)

It's a group of statements that together perform a specific task; and a block of code which only runs when it's called.

Why a function?

- ✓ **A piece of reusable code.**
- ✓ **Make the code shorter instead of repeating the same commands again.**
- ✓ **You can use it whenever you want!**

WHAT ARE THE TYPES OF FUNCTIONS?



PRE-DEFINED OR BUILT-IN FUNCTIONS

Function	Description	Argument type	Return type	Header file
sqrt	Square root	double	double	cmath
pow	Powers	double	double	cmath
abs	Absolute value for int	int	int	cstdlib
exit	End program	int	void	cstdlib
assert	Abort program with message	boolean	void	assert

EXAMPLES ON PRE-DEFINED FUNCTIONS

sqrt()

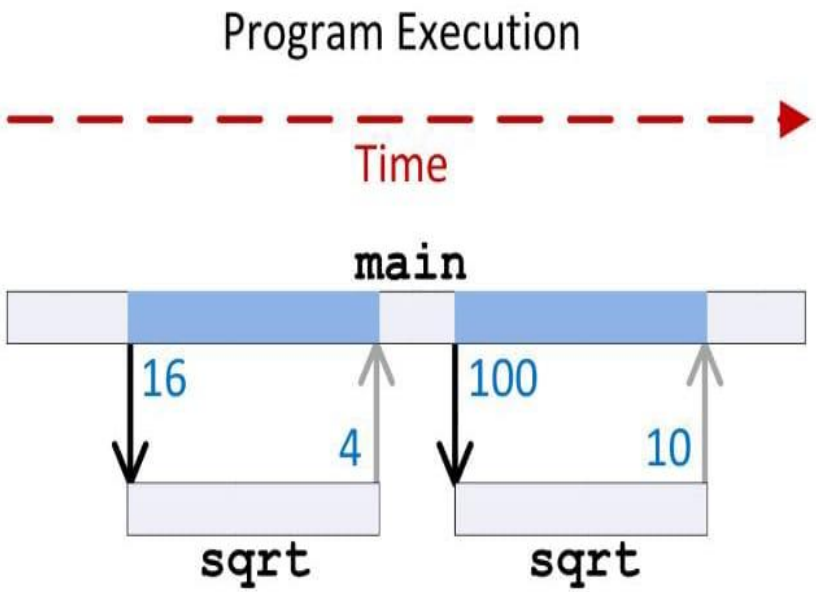
//one parameter

```
int main() {
    double value;

    //  Assign variable
    value = 16;

    //  Compute s square root
    double root = sqrt(value);

    //  Compute another
    root = sqrt(100);
}
```



If the caller code attempts to pass a parameter to the function that is incompatible with the type expected by the function, the compiler will issue an error.

```
cout << sqrt("16") << endl; // Illegal, a string is not a number
```

The compiler is able to determine that the above statement is illegal based on the additional information the preprocessor added to the code via the

```
#include<cmath>
```

EXAMPLES ON PRE-DEFINED FUNCTIONS

max()

//two parameters

Some functions take more than one parameter; for example, the C++ **max** function requires two arguments in order to produce a result. The **max** function selects and returns the larger of the two parameters. The **max** function is visualized in Figure 8.3. The **max** function could be used as

```
cout << "The larger of " << 4 << " and " << 7
     << " is " << max(4, 7) << endl;
```

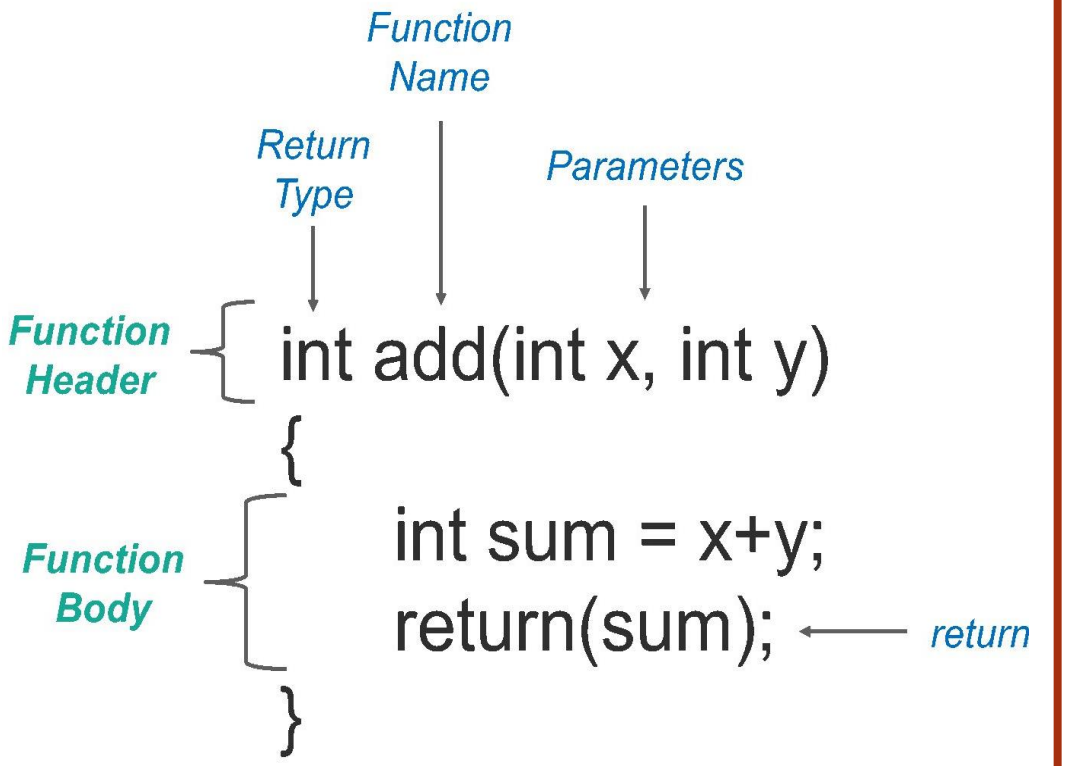
Notice that the parameters are contained in parentheses following the function's name, and the parameters are separated by commas.

From the caller's perspective a function has three important parts:

USER-DEFINED FUNCTIONS

- Void functions: do not have a return type
- Value-returning functions: have a data type
- To use these functions you need to:
  - Include the correct header file
  - Know the name of the function
  - Know the number of parameters, if any
  - Know the data type of each parameter
  - Know the data type of the value computed by the function, called the type of the function

USER-DEFINED FUNCTION'S SYNTAX



- The function is the standard unit of reuse in C++.
- The code that uses a function is known as calling code.
- Any C++ program should have at least one function which is the main().
- However, the program can be compiled without it, but it cannot be executed!

Now, suppose you need to compute the square root of a number. Would it be a good idea to write a function to make that?

EXAMPLES ON USER-DEFINED FUNCTIONS

A function with a welcome message

```
1  #include <iostream>
2  using namespace std;
3
4  void say_welcome() {
5       cout << "Welcome to E-JUST CS club!";
6        }
7
8  int main() {
9
10   say_welcome();
11 }
```

```
Welcome to E-JUST CS club!
[Program finished]
```

A function calculates a square's area

```
1  #include <iostream>
2  using namespace std;
3
4  int square_area(int x) {
5       int  area = x * x ;
6       return area;
7        }
8
9  int main() {
10
11   int a = square_area(4);
12   cout<<a;
13 }
```

```
16
[Program finished]
```

PASSING BY VALUE

The default parameter passing mechanism in C++ is classified as pass by value, also known as call by value.

This means the value of the actual parameter is copied to the formal parameter for execution.

```
1  #include <iostream>
2  using namespace std;
3  /*
4  * increment(x)
5  * Illustrates pass by value protocol.
6  */
7  void increment(int x) {
8     cout << "Beginning execution of
   increment, x = " << x << endl;
9     x++; // Increment x
10    cout << "Ending execution of increment,
   x = " << x << endl;
11 }
12 int main() {
13   int x = 5;
14   cout << "Before increment, x = " << x <<
   endl;
15   increment(x);
16   cout << "After increment, x = " << x <<
   endl;
17 }
```

```
Before increment, x = 5
Beginning execution of increment, x = 5
Ending execution of increment, x = 6
After increment, x = 5

[Program finished]
```

WHAT'S THE OUTPUT OF THIS CODE? (10 MINUTES)

```
1  #include <iostream>
2  using namespace std;
3
4  int circle_area(double r) {
5       const double pi = 3.14;
6       double  area = pi * r * r ;
7       return area;
8        }
9
10 int main() {
11
12   int a = circle_area(4);
13   cout<<a;
14 }
```

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  int triangle_hypotenuse(int x, int y) {
6       int  z = sqrt(x * x + y * y);
7       return z;
8        }
9
10 int main() {
11
12   triangle_hypotenuse(3,4);
13 }
```

WHAT'S THE OUTPUT OF THIS CODE? (10 MINUTES)

```
1  #include <iostream>
2  using namespace std;
3
4  int circle_area(double r) {
5       const double pi = 3.14;
6       double  area = pi * r * r ;
7       return area;
8        }
9
10 int main() {
11
12   int a = circle_area(4);
13   cout<<a;
14 }
```

50.24

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  int triangle_hypotenuse(int x, int y) {
6       int  z = sqrt(x * x + y * y);
7       return z;
8        }
9
10 int main() {
11
12   triangle_hypotenuse(3,4);
13 }
```

WHAT'S THE OUTPUT OF THIS CODE? (10 MINUTES)

```
1  #include <iostream>
2  using namespace std;
3
4  int circle_area(double r) {
5       const double pi = 3.14;
6       double  area = pi * r * r ;
7       return area;
8        }
9
10 int main() {
11
12   int a = circle_area(4);
13   cout<<a;
14 }
```

50.24

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  int triangle_hypotenuse(int x, int y) {
6       int  z = sqrt(x * x + y * y);
7       return z;
8        }
9
10 int main() {
11
12   triangle_hypotenuse(3,4);
13 }
```

5

CAN YOU SOLVE THIS PROBLEM? (10 MINUTES)

12. Write a function definition for a function called even that takes one argument of type `int` and returns a `bool` value. The function returns `true` if its one argument is an even number; otherwise, it returns `false`.

HINT

- To Know if the function is even or odd use %

HINT

```cpp
#include<iostream>
using namespace std;

bool odd_or_even(int num)
{
    if(num%2 == 0)
     return true;
    else
     return false;
}

int main()
{
    int x = 9;
    cout<<odd_or_even(x);

}
```

LET'S WRITE MORE THAN ONE FUNCTION! (15 MINUTES)

14. Write a function definition for a function `is_root_of` that takes two arguments of type *`int`* and returns a *`bool`* value. The function returns *`true`* if the first argument is the square root of the second; otherwise, it returns *`false`*.

LET'S WRITE MORE THAN ONE FUNCTION! (15 MINUTES)

- You can use Sqrt from cmath or check n*n

LET'S WRITE MORE THAN ONE FUNCTION! (15 MINUTES)

```cpp
#include<iostream>
using namespace std;

bool is_root(int x, int y)
{
    if(x*x == y)
     return true;
    return false;
}

int main()
{
    cout<<is_root(3,4)<<endl;
}
```

LET'S DO MORE! (25 MINUTES)

- create a function to print prime numbers between 1 and any number
- Take the number from the user

LET'S DO MORE! (25 MINUTES)

- The Prime number is the number who only divide by 1 and itself.

LET'S DO MORE! (25 MINUTES)

```cpp
#include<iostream>
using namespace std;



bool isPrime(int num){
    for (int i = 2; i < num; i++){
        if (num % i == 0){
            return false;
        }
    }
    return true;
}


int main(){
    int b;
    cin>>b;
    for (int i = 1; i <= b; i++){
        if (isPrime(i))
        {
            cout<<i<<endl;
        }
    }
    return 0;
}
```

LET'S DO MORE! (25 MINUTES)

How do you think we can optimize the performance of this algorithm ??

LET'S DO MORE! (25 MINUTES)

```cpp
#include<iostream>
using namespace std;



bool isPrime(int num){
    for (int i = 2; i <= num/2; i++){
        if (num % i == 0){
            return false;
        }
    }
    return true;
}


int main(){
    int b;
    cin>>b;
    for (int i = 1; i <= b; i++){
        if (isPrime(i))
        {
            cout<<i<<endl;
        }
    }
    return 0;
}
```